(19)日本国特許庁(JP)　(12)公開特許公報（A）　(11)特許出願公開番号

特開2000－285000
(P2000－285000A)
(43)公開日　平成12年10月13日(2000.10.13)

(51)Int.Cl.⁷		識別記号	FI			テーマコード(参考)
G06F	12/00	542	G06F	12/00	542 K	5B035
	12/02	510		12/02	510 A	5B058
		530			530 B	5B060
G06K	17/00		G06K	17/00	D	5B082
	19/07			19/00	N	

審査請求　未請求　請求項の数9　OL　(全13頁)

(21)出願番号　特願平11-92562

(22)出願日　平成11年3月31日(1999.3.31)

(71)出願人　000005821
松下電器産業株式会社
大阪府門真市大字門真1006番地

(72)発明者　廣田　照人
大阪府門真市大字門真1006番地　松下電器産業株式会社内

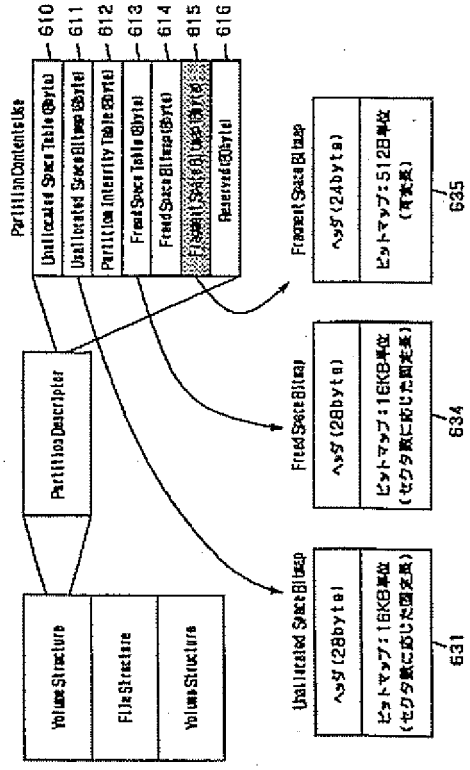
(72)発明者　前田　卓治
大阪府門真市大字門真1006番地　松下電器産業株式会社内

(74)代理人　100097445
弁理士　岩橋　文雄　(外2名)

最終頁に続く

(54)【発明の名称】不揮発メモリのファイルシステム

(57)【要約】
【課題】　デジタル著作物のデータ構造を、記録媒体及びデータの特性に合わせた最適な構造を取ることにより、高速アクセス性と省メモリ性を実現したデジタル著作物記録媒体を提供する。
【解決手段】　デジタル著作物のデータを記録媒体への書き込む際のデータサイズを、消去サイズと等しく行い、記録媒体のデータの消去は書き込み時だけでなく、書き込み時以外の空き時間にも行い、それ以外のデータはセクタサイズで書き込みを行う。デジタル著作物のデータを識別する属性をもうけることで、著作物データの上書き、追記といった改変処理を防止すると共に、著作物データに関し不要な機能を実装しないことで、省メモリを実現する。



1

【特許請求の範囲】

【請求項１】　デジタル著作物データをファイルとして扱うファイルシステムの論理ブロックを、ファイルのデータ領域は不揮発メモリの消去サイズと同じサイズのＡＶブロック、ファイル名、ファイル属性を管理するファイルエントリ、全ブロックの空き領域の管理をする管理情報領域はセクタサイズと同一サイズの管理ブロックとに２種類の論理ブロック構造を持つファイルシステム。

【請求項２】　デジタル著作物データをファイルとして扱うファイルシステムが、不揮発メモリの論理ブロックを、書き込みを行うために消去処理が必要なブロックをフリーリストとして管理し、論理ブロックが既に消去済みであるブロックを消去済みフリーリストとして管理する構造を持つファイルシステム。

【請求項３】　分割・結合を行うデジタル著作物データのファイルに対してＡＶ属性を付けることで、ＡＶ属性の場合は、分割・結合処理部は動作し、追記・上書き処理部を未動作にするＡＶ属性判定機能を備えるファイルシステム。

【請求項４】　不揮発メモリの論理ブロックと物理ブロックの対応を変更し、ファイルの論理ブロックの配置を連続的に割り当てる機能を備えるファイルシステム。

【請求項５】　前記管理ブロックのデータの割り当てを、前記ＡＶブロックから割り当てを行い、前記管理ブロックに空きがある場合は、前記ＡＶブロックとして連続配置を行う機能を備える請求項１記載のファイルシステム。

【請求項６】　不揮発メモリの全論理ブロックを管理するボリューム管理領域のコピーを２個以上備えるファイルシステム。

【請求項７】　前記ボリューム管理領域と前記ボリューム管理領域のコピーに、識別番号を備える請求項６記載のファイルシステム。

【請求項８】　前記ボリューム管理領域を前記不揮発メモリの消去サイズの整数倍である請求項７記載のファイルシステム。

【請求項９】　前記不揮発メモリは、前記不揮発メモリをアクセスする装置との間で、相互認証を行うアクティブ素子を有することを特徴とする請求項１～８のいずれか１項に記載のファイルシステム。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は、デジタル著作物を記録するための記録媒体のデータ構造に関し、また、当該記録媒体へのアクセス装置に関する。

【０００２】

【従来の技術】近年、マルチメディア・ネットワーク技術の発展により、デジタル著作物である音楽等のコンテンツがインターネット等を通じて配信されるようになり、自宅に居ながらにして世界中の音楽等に接することが可能となってきた。また、このような音楽コンテンツ等は、不揮発性の半導体メモリ、フラッシュメモリ等の記録媒体に記録することも可能である。半導体メモリ等の記録媒体に記録された音楽コンテンツは、例えば、携帯型の音楽再生装置により読み出され再生される。

2

【０００３】ところで、記録媒体にデータを記録する場合、内容的にまとまりのあるデータをその単位で扱いやすくするためにファイルの概念を用い、記録媒体にはデータのまとまりであるファイルと、ファイルを管理する管理情報とを対にして記録することができる。管理情報は、例えばファイルについてのアクセス可否等の属性やサイズやファイルの位置等を示す情報であり、この管理情報を参照することによりファイルへアクセスすることができる。このファイルとして扱う機構をファイルシステムと呼ぶ。

【０００４】

【発明が解決しようとする課題】しかしながら、従来、コンテンツを記録する記録媒体に用いられるフラッシュメモリは、コンピュータ用の補助記憶用として使用されることが想定されていた。フラッシュメモリに書き込む際にフラッシュメモリの特性に最適化されたデータ構造ではなく、互換性の為にＪＩＳ－Ｘ－０６０５規格のＦＡＴと呼ばれる形式のデータ構造を利用しているため、高速に書き込むことができなかった。携帯型の音楽再生装置では、使用者は朝の出勤や登校前の時間にフラッシュメモリへ書き込むことが想定される為、できるだけ短時間で書き込みが完了することが要求される。つまり、インターネットからダウンロードしたコンテンツをフラッシュメモリにできるだけ高速に書き込むことが要求される。

【０００５】また、民生用の携帯型音楽再生装置で利用する場合は、フラッシュメモリへのデータ書き込み中の電源断等の非常事態が発生してもフラッシュメモリ内部のデータができるだけ破壊されないようなデータ構造を取る必要がある。

【０００６】そこで、本発明は、このような要請に鑑みてなされたものであり、不揮発性の半導体メモリ、フラッシュメモリの利用面、安全性とにおいて最適なデータ構造でデジタル著作物が記録されたデジタル著作物記録媒体、及びそのアクセス装置を提供することを目的とする。

【０００７】

【課題を解決するための手段】上記課題を解決するために本発明に係るデジタル著作物媒体は、デジタル著作物データをファイルとして扱うファイルシステムの論理ブロックを、ファイルのデータ領域は不揮発メモリの消去サイズと同じサイズのＡＶブロック、ファイル名、ファイル属性を管理するファイルエントリ、全ブロックの空き領域の管理をする管理情報領域はセクタサイズと同一サイズの管理ブロックとに２種類の論理ブロック構造を

持つことを特徴とする。

【０００８】また、前記管理ブロックのデータの割り当てを、前記ＡＶブロックから割り当てを行い、前記管理ブロックに空きがある場合は、前記ＡＶブロックとして連続配置を行う機能を備えることを特徴とする。

【０００９】また、デジタル著作物データをファイルとして扱うファイルシステムが、不揮発メモリの論理ブロックを、書き込みを行うために消去処理が必要なブロックをフリーリストとして管理し、論理ブロックが既に消去済みであるブロックを消去済みフリーリストとして管理する構造を持つことを特徴とする。

【００１０】また、分割・結合を行うデジタル著作物データのファイルに対してＡＶ属性を付けることで、ＡＶ属性の場合は、分割・結合処理部は動作し、追記・上書き処理部を未動作にするＡＶ属性判定機能を備えることを特徴とする。

【００１１】また、不揮発メモリの論理ブロックと物理ブロックの対応を変更し、ファイルの論理ブロックの配置を連続的に割り当てる機能を備えることを特徴とする。

【００１２】また、不揮発メモリの全論理ブロックを管理するボリューム管理領域のコピーを２個以上備え、前記ボリューム管理領域と前記ボリューム管理領域のコピーに、識別番号を備え、前記ボリューム管理領域を前記不揮発メモリの消去サイズの整数倍である特徴を備える。

【００１３】

【発明の実施の形態】以下、本発明に係るデジタル著作物記録媒体のデータ構造について、図面を用いて説明する。

【００１４】＜１．メディアカードについて＞携帯型の録再装置（以下、半導体プレーヤ）では、デジタル著作物記録媒体（以下、メディアカード）に記録された音楽データを再生する。メディアカードは、何度も繰り返して書き込みが行える書き換え可能な不揮発メモリであり、容量は６４ＭＢある。なお、メディアカードは、厚さ２ｍｍ、縦横２ｃｍ四方程度の形状で、フラッシュメモリの他に、相互認証、フラッシュメモリのアクセス制御、待避処理等の機能をもつアクティブ素子を有する。

【００１５】メディアカードは、フラッシュメモリの特性から、最小の読み書き単位はセクタサイズ（５１２バイト）で可能であるが、書き込む場合は消去が必要で、その大きさはクラスタサイズ（１６ＫＢ）単位である。したがって、それ以下の大きさ、例えば５１２バイトの書き込みが発生する場合には、１６ＫＢ残りの部分とともに書き込みを行う待避処理を行う必要がある。

【００１６】この待避処理の動作を図１に従って説明する。

1.１６ＫＢ以下（５１２バイト）の書き込み要求が発生（Ｓ１０１）

2.書き込みを行うクラスタのデータ（１６ＫＢ）をメモリ上に退避する（Ｓ１０２）

3.メモリ上で書き込み部分のデータ（５１２バイト）を変更する（Ｓ１０３）

4.メディア上で書き込み位置のクラスタのデータ（１６ＫＢ）を消去する（Ｓ１０４）

5.メモリ上のデータ（１６ＫＢ）をフラッシュメモリに書き込む（Ｓ１０５）

したがって１６ＫＢ以下の書き込みを行う場合、いったんメモリ上に退避してから書き込むことになり、実際に書き込みを行いたい部分以外でも書き込みが生じることになる。そのため、１６ＫＢ以下のデータを複数回書き込むよりも、データの退避が起こらない１６ＫＢのデータを一括して書き込んだ方が高速に書き込みが行える。また、メディアカードは30万回書き込むとそれ以上書けなくなるという書き込み回数に対する寿命があることから、書き込み回数をできる限り低減する必要がある。その点から１６ＫＢのデータを一括して書き込むことが必要となる。

【００１７】＜２．ＵＤＦについて＞図２はフラッシュメモリに格納される論理フォーマット、ＵＤＦ（ＵｎｉｖｅｒｓａｌＤｉｓｋＦｏｒｍａｔ）の概略図である。ＵＤＦでは、ボリュームの先頭と最後の各257セクタにVolume Structure２００が存在し、ボリューム全体の管理情報を格納している。後半のVolume Structure２２０のほとんどは前半のVolume Structure２００のコピーとなっており、前半のVolume Structure２００が破壊されたときの障害復帰用として存在している。2つのVolume Structure２００，２２０の間には、実際にファイルのデータなどを格納するFile Structure２１０が存在する。

【００１８】図３はFile Structure３００の概略図である。File Structure３００には、その領域であるPartitionの情報を格納するFile Set Descriptor３０１、File SetDescriptor３０１の終端を示すTerminating Descriptor３０２、領域管理のためのSpace Bitmapを格納するSpace Bitmap Descriptor３０３、ファイルの情報を格納するFile Entry３０４、ディレクトリ中のFileの情報を格納するFile Identifier Descriptor３２０、ファイルのデータなどが格納される。

【００１９】＜３．Volume Structureについて＞ＵＤＦではVolume Structure２００、２２０のために256個のセクタ領域を2つ確保している。また、Volume Structure２００，２２０とFile Structure２１０の間に15セクタのReserved領域を確保している。半導体プレーヤでは領域が限られているため、不必要なセクタは極力省く必要がある。

【００２０】また、Volume Structure２００、２２０の一部分はファイルの作成、削除などの際に更新される必要がある。ここで変更されるデータはVolume Structure

２００、２２０の一部分であるが、上述したメディアカードの特性により、データの書き込みは１６ＫＢ単位で行ったときが最速になるため、Volume Structure２００、２２０の書き込みも１６ＫＢ単位で行うことが望ましい。

【００２１】これらの点から、Volume Structure２００、２２０の有用な部分のみを抽出し、１６ＫＢのクラスタ内に格納する方法が有効である。しかしながらVolume Structure２００，２２０で有用な部分は１６ＫＢも存在していないため、File Structure２１０の中に記述されているPartitionの管理情報の一部をここに含めることで、１６ＫＢを有効に利用することができる。

【００２２】図４に今回発明したボリュームフォーマットの概略図を示す。このフォーマットではＵＤＦのフォーマットと同様、ボリュームの先頭と最後にボリュームの管理情報を格納するVolume Structure４００，４４０が存在し、その間にはファイルのデータなどを格納するFile Structure４２０が存在する。

【００２３】＜３．１複数のVolume Structureの使用について＞メディアカードの特性として、書き込み回数の上限が決まっているため、同一セクタに書き込みが集中することは避けなければならない。そのため同じ構造を持ったVolume Structure４００，４４０を2つ用意し、管理情報の更新が行われるごとに交互にVolume Structure４００，４４０の更新を行う。更新の際には2つのVolume Structure４００、４４０で共通に管理しているID値をインクリメントしていき、最新の管理情報が常に大きな値を保持するようにしておく。このようにすることで、Volume Structure４００、４４０への管理情報の書き込みが2クラスタに分散され、メディアカードの寿命を延ばすことができる。どちらかのVolume Structure４００、４４０が壊れた場合には、壊れていないVolume Structure４００、４４０を利用し、両方とも壊れていない場合は、ID値の大きい方を利用する。

【００２４】さらにVolume Structure４００，４４０の個数は2つだけではなく、それ以上の個数を用意することも可能である。そうすることで、さらに障害に強くなり、書き込み回数も分散化されることになる。

【００２５】＜３．２ File Structureの一部をVolume Structureと同一クラスタに含めることについて＞ＵＤＦではSpace Bitmap４２３をFile Structure４２０内に配置している。Space Bitmap４２３は、メディアカード内のセクタの使用、未使用を示すデータであるため、ファイルの作成、更新などの際に頻繁に変更されるものであり、録音中に電源が落ちるなどの突発的な事態に対して録音済みの曲の保護等を考えた場合、曲の録音完了時ごとにメディアに書き込みを行う必要がある。そうした場合、同一セクタに対する書き込み回数が多くなり、前述したメディアカードの書き込み回数の制限から、そのセクタの寿命が短くなる問題がある。

【００２６】そこで、Space Bitmap４２３などのFile Structure４２０に含まれていた管理情報をVolume Structure４００の後に配置することでVolume Structure４００と同一のクラスタとし、前述したVolume Structure４００，４４０を複数持つ場合と同様にして、書き込みセクタの分散化を行って書き込み寿命を延ばす。

【００２７】また、Volume Structure４００、４４０と同じ１６ＫＢの領域に収めることで高速な書き込みを行う。

【００２８】＜４．ＵＤＦの領域管理方法について＞図５はＵＤＦの領域管理方法の概略図である。ＵＤＦではVolume Structure５００中にあるPartition Descriptor５０１で、Partitionの領域管理の方法を示している。Partition Descriptor５０１内のPartition Contents Use５０２フィールドには、Partitionで使用する領域管理のBitmap領域などのアドレスと大きさが記述される。実際の領域管理はそのアドレスの先にあるBitmapあるいはTableで行われる。ＵＤＦでは領域管理方法としてSpace BitmapとSpace Tableをサポートしているが、両方を同時に使用してはならないという制限事項がある。Space Bitmap５１１、５１４は管理を行う全セクタ数と同じ数のビット情報を持ち、ビットの0,1で領域の使用、未使用を管理する方法である。Space Table５１０，５１３は、未使用領域を開始位置と領域のサイズの組み合わせで記述し、管理する方法である。Unallocated Space５１０，５１１は未使用領域で、すぐに書き込みが可能な領域を管理するものである。Freed Space５１３，５１４は未使用領域であるが、書き込みの前に前処理が必要で、すぐには書き込みが行えない領域を管理するものである。

【００２９】＜５．半導体プレーヤの領域管理方法について＞半導体プレーヤにおいてメディアカードにデータを書き込む場合、すでにデータが書かれている領域では、一度データを消去してから書き込まなければならない。５１２バイトの書き込み時間が２００μ秒であるのに対して、１６ＫＢあたりのデータ消去時間は約２ｍ秒かかるため、実際に書き込みにかかる時間の大半をデータ消去時間が占めることになる。この問題を解消するため、データが消去されていない未使用領域を管理するSpace Bitmapと、データ消去済みの未使用領域を管理するSpace Bitmapの2つにより領域管理を行う。未使用領域をデータの有無で別々に管理することにより、空き時間を利用してデータを先行消去することが可能となり、データの高速な書き込みが実現できる。

【００３０】また、前述したように小さなサイズのデータを複数回書き込むよりも、１６ＫＢ単位のデータを一括して書き込む方が高速に書き込むことができることから、管理領域用のセクタサイズ（５１２バイト）とは別に、書き込み用に大きなサイズ（１６ＫＢ）のクラスタを用意しそれぞれ別のSpace Bitmapで領域の管理を行

う。

【００３１】上記の２つの領域を管理するSpace Bitmapの概略図を図６に示す。この図に示すように、未使用領域におけるデータの有無を管理するためにUnallocated Space Bitmap６１１と、Freed Space Bitmap６１４を用い、高速書き込み用にサイズの異なる領域を管理するためにUnallocated Space Bitmap６１１とFragment Space Bitmap６１５を用いている。Unallocated Space Bitmap６３１は未使用でデータ消去済みである領域を管理し、1ビットに対応する大きさは１６ＫＢである。Freed Space Bitmap６３４は未使用であるがデータが未消去の領域を管理し、1ビットに対応する大きさは１６ＫＢである。Fragment Space Bitmap６３５はFile Entryなどのファイル管理情報の領域を管理し、1ビットに対応する大きさは５１２バイトである。

【００３２】図７にUnallocated Space Bitmap６３１、Freed Space Bitmap６３４の構成を示す。これらは同一の構成をしており、ＵＤＦのSpace Bitmap Descriptorの拡張となっている。Descriptor Tag７０１フィールドはこのを識別するタグである。Number of Bits７０２はビットマップのビット数であり、管理する領域の全クラスタ数を示す。Number of Bytes７０３はビットマップで使用するバイト数を示す。ここまではＵＤＦのSpace Bitmap Descriptorと同じである。Allocation Starting Position７０４は、割り当て要求（あるいは初期化要求）が発生したときに空き領域を探索する際の開始クラスタ番号を示す。この数値は最後に割り当てを行ったクラスタの番号を格納しておき、次回に割り当てるのはその続きから探索して最初に発見された空き領域となる。このように管理することで割り当てクラスタがPartition全体に分散化され、書き込み回数が均等化するため、メディアカードの寿命を延ばすことができる。

【００３３】図８にFragment Space Bitmap６３５の構成を示す。Descriptor Tag８０１フィールドはこのを識別するタグである。Bitmap Size８０２フィールドは1クラスタ中のセクタ数で、1つのビットマップのビット数である。半導体プレーヤの場合、セクタサイズが５１２バイト，クラスタサイズが１６ＫＢなので、Bitmap Sizeは32となる。Number of Bitmaps８０３フィールドは、このDescriptor内に格納されているビットマップの数である。ここまでが、Descriptorのヘッダ領域である。以下の領域では、Fragment Position８０４とBitmap８０５の組み合わせが1つ以上で状況に応じて複数個存在する。Fragment Position８０４はBitmap８０５が示す領域がどこから始まる領域なのかを示すものであり、Bitmap８０５の先頭セクタの論理セクタ番号が格納される。Bitmap８０５は実際に領域管理を行うビットマップであり、Bitmap Size８０２のビット数を持つ。このFragment Space Bitmap６３５は、Unallocated Space Bitmap６３１が１６ＫＢ単位で領域管理を行うのに対し、５１２バイト単位の領域管理を行う。

【００３４】次に、図９のフロー図を用いて、上記の3つのSpace Bitmap間の割り当て関係を説明する。

1.ファイルのデータを書き込む場合は、Unallocated Space Bitmapから割り当てを行う（Ｓ９０１）

2. Unallocated Space Bitmapに空き領域のない場合、空き時間がある場合はFreed Space Bitmapからデータを初期化して空き領域を作成する（Ｓ９０２）

3.ファイルのデータを削除する場合は、空きとなった領域をFreed Space Bitmapが管理する（Ｓ９０３）

4. File Entryなどのファイル管理情報を作成する場合は、Fragment Space Bitmapから割り当てを行う（Ｓ９０４）

5. Fragment Space Bitmapに空き領域がない場合は、Unallocated Space Bitmapから割り当てを行う。Unallocated Space Bitmapにも空き領域がない場合は、Freed Space Bitmapからデータを初期化して割り当てを行う（Ｓ９０５）

6. File Entryなどのファイル管理情報を削除する場合は、空きとなった領域をFragment Space Bitmapが管理する（Ｓ９０６）

7. Fragment Space Bitmapで空き領域が32セクタ以上になれば、それらを1つのクラスタに結合し、Freed Space Bitmapの空き領域として管理する。（Ｓ９０７）

次に、図１０を用いて、Fragment Space Bitmapについて詳しく説明する。

1. Fragment Space BitmapはUnallocated Space Bitmapから１６ＫＢの領域を割り当ててもらい、その領域を５１２バイトずつ管理情報用に割り当てる（Ｓ１００１）

2.現在保持している領域がすべて管理情報に割り当てられると、そのSpace Bitmap情報は必要なくなるので消去し、新たにUnallocated Space Bitmapから１６ＫＢ割り当ててもらう（Ｓ１００２）

3.管理領域が解放された場合は、現在Space Bitmapを持っていればそのSpace Bitmapを変更する（Ｓ１００３）

4.上記でSpace Bitmapを持っていない場合は新たにSpace Bitmapを作成する（Ｓ１００４）

5.領域が離散的に解放され、複数のFragment Space Bitmapを保持しなければならなくなった場合は、SB2の保持するfile1のFile Entryと、SB3の保持するfile2のFile EntryをSB1の空き領域にコピーする。次にすべての領域が空きとなったSB2とSB3をFreed Space Bitmapに解放する。その結果、保持するデータはSB1のみとなり、データ量が低減できる。（Ｓ１００５）

領域管理の例として、図１１にファイル作成時の領域割り当て方法を示す。

1.ファイルの情報を記述するFile Entryを作成するために、File Entry用の領域をFragment Space Bitmapから１セクタ（５１２バイト）割り当てる（Ｓ１１０１）

2. Fragment Space Bitmapに割り当てられる領域が存在しない場合、Unallocated Space BitmapからFragment Space用に1クラスタ（１６ＫＢ）割り当てる（Ｓ１１０２）

3.上記でUnallocated Space Bitmapに割り当て領域が存在しない場合、Freed Space BitmapからFragment Space用に1クラスタ（１６ＫＢ）初期化して割り当てる（Ｓ１１０３）

4.上記でFreed Space Bitmapに割り当て領域が存在しない場合、メディアがフルの状態であるの新たにファイルを作成することはできない（Ｓ１１０４）

5. File Entryを作成する（Ｓ１１０５）

6.ファイルのデータ領域（１６ＫＢ）をUnallocated Space Bitmapから割り当てる（Ｓ１１０６）

7. Unallocated Space Bitmapに割り当てられる領域が存在しない場合、Freed Space Bitmapから必要な領域を初期化して割り当てる（Ｓ１１０７）

8.上記でFreed Space Bitmapに割り当てられる領域が存在しない場合、それ以上のデータを書き込むだけの領域が存在しないので、ファイルにデータを書き込む作業を終了する（Ｓ１１０８）

＜６．物理アドレスと論理アドレスの対応表について＞

メディアカードでは、物理的に存在するすべてのデータ空間にデータを書き込めるのではなく、論理セクタと呼ばれる領域にのみデータを書き込むことができる。論理セクタのアドレスである論理アドレスは、物理的に並んで存在しているとは限らない。これは、メディアの一部が破損し読み書きが行えなくなった空間を、他の代替空間に置き換える際に論理アドレスと物理アドレスの対応を書き換えることによる。すなわちメディアカードでは、物理的に不連続な領域を、物理アドレスと論理アドレスの対応表を書き換えることにより、論理的に連続なセクタであると見せかけている。

【００３５】一方、今回のファイルシステムでは、ファイルの割り当て領域を示すためにAllocation Descriptorを使用している。このAllocation Descriptorは、割り当て開始位置と割り当てサイズの組み合わせで領域を表現しているため、割り当て領域が離散的になっているとそれを記述するAllocation Descriptorの個数が増加し、ファイル管理情報のオーバーヘッドが増大することになる。そのため、割り当て領域は可能な限り連続領域とする必要がある。

【００３６】この問題を解決する方法として、物理アドレスと論理アドレスの対応表を書き直す方法がある。図１２に対応表を書き直す場合の例を示す。この例では、物理アドレス0と2にfile1のデータが存在し、物理アドレス1にfile2のデータが存在する。対応表１２０１では、物理アドレス１２０３と論理アドレス１２０２を同じ値としているので、論理アドレス上でもfile2のデータがfile1のデータに挟まれている。そこで対応表を書き換えた場合、物理アドレス１２１３ではデータの位置が変わらないが、論理アドレス１２１２ではデータが連続している。このようにすると、１つのAllocation Descriptorで領域を表現することが可能である。

【００３７】前述のファイルシステムでは、領域管理としてSpace Bitmapを用いる方法を示してきた。Space Bitmapの利点としてはサイズが管理する領域の大きさに比例した固定長となることがあげられるが、逆に管理する領域が大きくなったときに大きなBitmapが必要となることが欠点となる。それに対しSpace Tableは、管理する領域が連続である場合効率的に管理ができ、離散的である場合に管理情報が増加する。そこで、上記の手法を用いて管理領域を連続にした場合、Space Tableを用いた方が効率的に管理が行える。

【００３８】＜７．ＡＶ属性について＞次に、デジタル著作物の編集を行う際のファイルの動作について説明する。文字などのファイルを編集する場合、そのファイルの一部分を変更し、上書きや追記などを行って、メディアカードに書き込む。しかし、音楽データや動画データは、録音する場合など編集するサイズが未定である為、そのファイルを直接編集することはない。例えばＭＤ（ＭｉｎｉＤｉｓｃ）の場合であれば、まず録音用に新規に１つのファイル（＝トラック）を作成する。その後、音楽と音楽を繋げて１曲にする場合は、新規に作成したファイルと別のファイルと結合して１つのファイルにする。また、音楽と音楽の間を分割する場合は、１つのファイルを２つのファイルに分割する操作を行う。このように音楽データは、文字データとは性質が異なるため、ファイルとしての編集操作方法が大きく異なる。

【００３９】そこで、ファイルシステムがあらかじめ当該ファイルが、分割・結合を行う必要があり上書き・追記が不要である音楽データか、分割・結合の必要はないが上書き・追記が必要である文字データかを判定できるようにし、この判定方法として、音楽データや動画データのファイルにはFile EntryにＡＶ属性を示す情報を入れ、それ以外の文字データにはＡＶ属性の情報を入れず、ファイルシステムは当該ファイルのFile EntryのＡＶ属性を参照することで判定できるようにする。このことにより、ファイルシステムは当該ファイル用の機能として、分割・結合・上書き・追記の全ての機能を準備しなくても、音楽データか文字データかによって必要最小限の機能だけを準備すればよいことになり、携帯型の録再装置を考えた場合、全ての機能を準備するのに比べメモリ量を削減することができる。

【００４０】以上、本発明に係るデジタル著作物記録媒体のデータ構造、及び当該記録媒体へのアクセス装置のファイルシステムについて、実施の形態に基づいて説明したが、本発明はこれらの実施の形態に限られないことは勿論である。即ち、

（１）デジタル著作物媒体として、フラッシュメモリを

用いたメディアカードであるとしたが、不揮発性メモリであれば、バブルメモリ等その他の不揮発性メモリを用いてもよい。

（２）フラッシュメモリの消去サイズを１６ＫＢと、セクタサイズを５１２バイトとしたが、このサイズに限定されるものではない。

（３）ファイルシステムとしてＵＤＦを拡張したが、ＮＴＦＳや独自のファイルシステムを用いて、本発明内容の拡張を行ってもよい。

（４）Volume Structureをメディアカードの先頭と最後に配置したが、メディアカード内であればどこでもよい。

（５）ＡＶ属性は、ファイルシステムのFile Entryにあるとしたが、ＡＶ属性の場所は、ＵＤＦの拡張属性としてもよいし、アプリケーションがファイルシステムにあらかじめＡＶ属性を知らせてもよい。

【００４１】

【発明の効果】以上の説明からも明らかなように、本発明に係るデジタル著作物記録媒体は、フラッシュメモリへのデータの書き込みを消去サイズ以上の大きさで行うため、高速に書き込みができ、消去処理をあらかじめ行うことで、データの書き込み時に消去せずに高速に書き込みできる。

【００４２】また、ファイルの配置情報を管理するブロック領域を、メディアカード内の論理ブロックと物理ブロックの対応を変更することで連続割り当てすることで最小化できる。

【００４３】また、ＡＶ属性を設けることでファイルシステムの必要最小限の機能の実装で済むため、携帯型で必須である省メモリ化が実現できる。

【図面の簡単な説明】

【図１】本発明の一実施の形態におけるメディアカード内のデータの待避処理動作を示す図

【図２】本発明の一実施の形態におけるＵＤＦの構成図

【図３】本発明の一実施の形態におけるFile Structureの構成図

【図４】本発明の一実施の形態におけるボリュームフォーマットの構成図

【図５】本発明の一実施の形態におけるＵＤＦの管理領域の構成図

【図６】本発明の一実施の形態におけるSpace Bitmapの構成図

【図７】本発明の一実施の形態におけるUnallocated Space Bitmap、Freed Space Bitmapの構成図

【図８】本発明の一実施の形態におけるFragment Space Bitmapの構成図

【図９】本発明の一実施の形態におけるSpace Bitmap間の割り当て動作フロー図

【図１０】本発明の一実施の形態におけるFragment Space Bitmapの割り当て動作フロー図

【図１１】本発明の一実施の形態におけるファイル作成時の領域割り当て動作フロー図

【図１２】本発明の一実施の形態におけるメディアカード内の論理ブロック、物理ブロック変更動作図

【符号の説明】

２００，２０２，４００，４４０，５００，５４０　ボリューム構造(VolumeStructure)

２０１，３００，４２０，５２０　ファイル構造(File Structure)

３０１，４２１　ファイル集合記述子(File Set Descriptor)

３０２，４２２　終端記述子(Terminating Descriptor)

３０３　空間ビットマップ記述子(Space Bitmap Descriptor)

３０４　ルートのファイルエントリ(File Entry(root))

３０５　ルートのディレクトリデータ(Directory Data(root))

３０６　ファイル1のファイルエントリ(File Entry(file1))

３０７　ファイル1のデータ(Data(file1))

３０８　ファイル2のファイルエントリ(File Entry(file2))

３０９　ファイル2のデータ(Data(file2))

３２０　親ディレクトリのファイル識別記述子(File Identifier Descriptor(parent))

３２１　ファイル1のファイル識別記述子(File Identifier Descriptor(file1))

３２２　ファイル2のファイル識別記述子(File Identifier Descriptor(file2))

４２３　空間ビットマップ(Space Bitmap)

４２４　予備(Reserved)

５０１　区画記述子(Partition Descriptor)

５０２　区画内容用(Partition Contents Use)

５０３　ヘッダ(Unallocated Space Bitmap)

５０４　ビットマップ(Unallocated Space Bitmap)

５１０，６１０　未割付け空間テーブル(Unallocated Space Table)

５１１，６１１，６３１　未割付け空間ビットマップ(Unallocated Space Bitmap)

５１２，６１２　区画保全テーブル(Partition Integrity Table)

５１３，６１３　割付け可能空間テーブル(Freed Space Table)

５１４，６１４，６３４　割付け可能空間ビットマップ(Freed Space Bitmap)

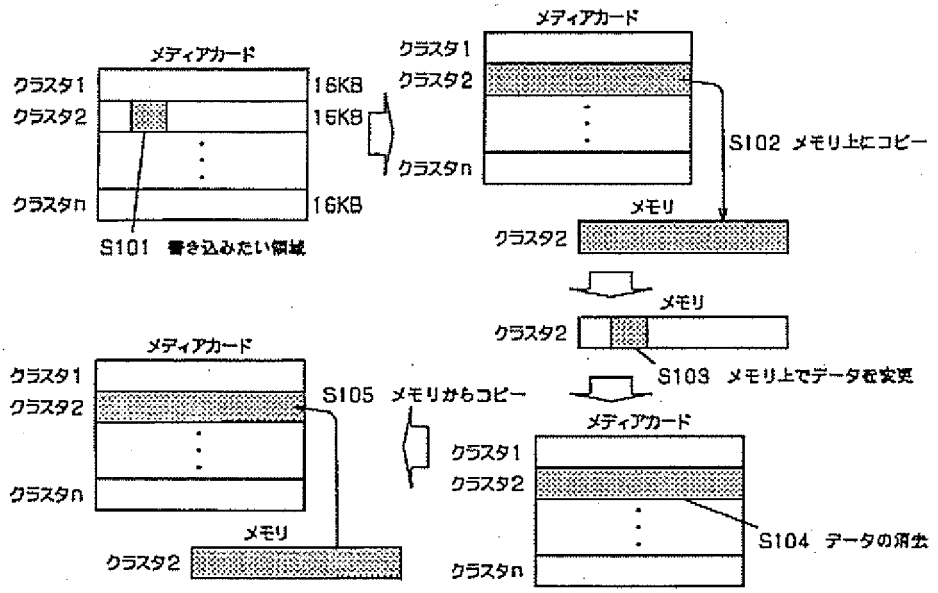
５１５，６１６　予備(Reserved)

６１５，６３５　フラグメント空間ビットマップ(Fragment Space Bitmap)

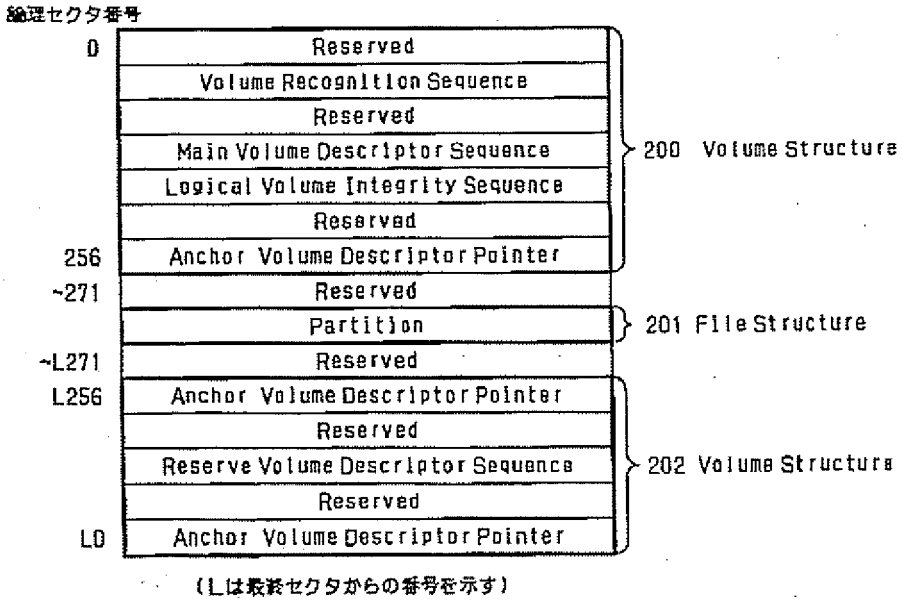
７０１，８０１　記述子タグ(Descriptor Tag)

７０２　ビット数(Number of Bits)
７０３　バイト数(Number of Bytes)
７０４　割付け開始位置(Allocation Starting Position)
７０５，８０５　ビットマップ(Bitmap)
８０２　ビットマップ長(Bitmap Size)
８０３　ビットマップ数(Number of Bitmaps)
８０４　フラグメント開始位置(Fragment Position)
１２０１，１２１１　対応表
１２０２，１２１２　論理ブロック
１２０３，１２１３　物理ブロック

【図１】

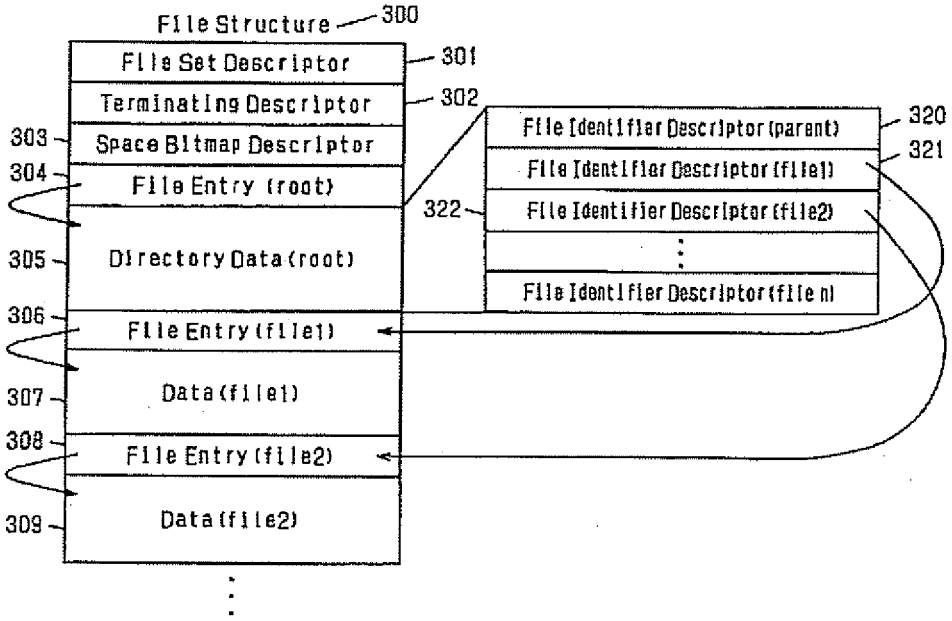


【図２】

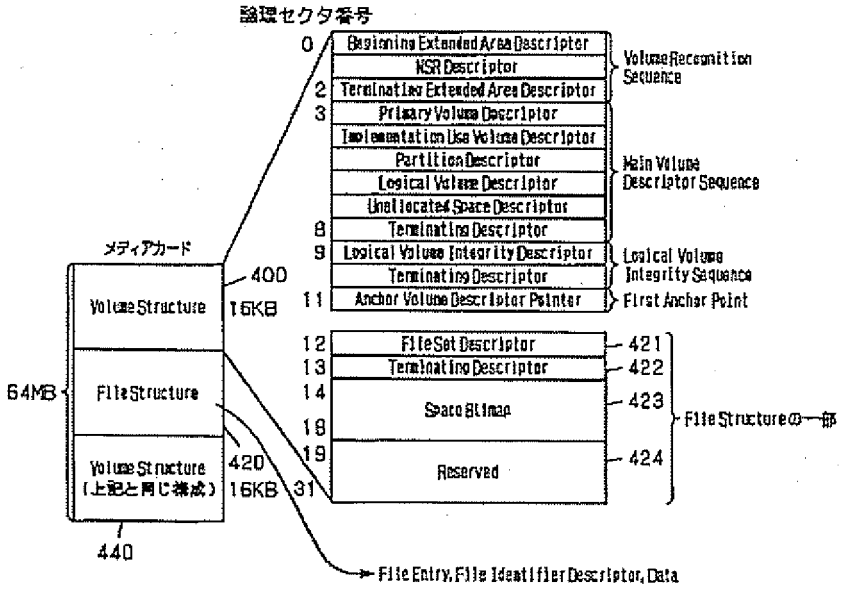


（Lは最終セクタからの番号を示す）

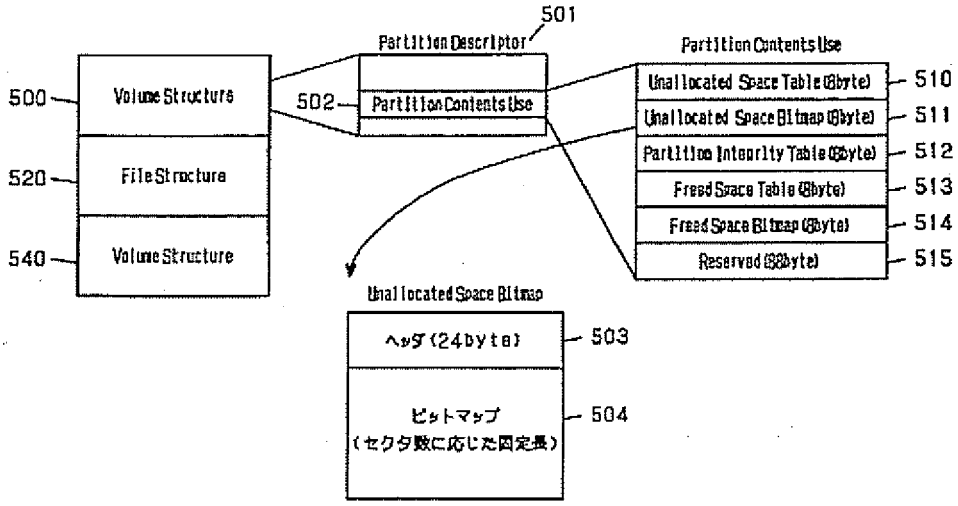
【図３】


File Structure
300
File Set Descriptor
301
Terminating Descriptor
302
303
Space Bitmap Descriptor
304
File Entry (root)
305
Directory Data (root)
306
File Entry (file1)
307
Data (file1)
308
File Entry (file2)
309
Data (file2)
322
File Identifier Descriptor (parent)
320
File Identifier Descriptor (file1)
321
File Identifier Descriptor (file2)
File Identifier Descriptor (file n)

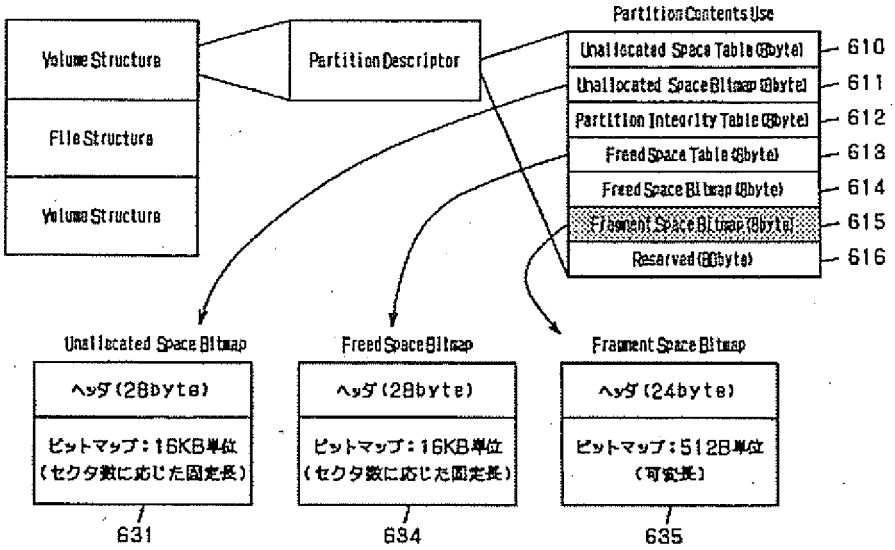
【図４】


論理セクタ番号
0
Beginning Extended Area Descriptor
NSR Descriptor
2
Terminating Extended Area Descriptor
Volume Recognition Sequence
3
Primary Volume Descriptor
Implementation Use Volume Descriptor
Partition Descriptor
Logical Volume Descriptor
Unallocated Space Descriptor
8
Terminating Descriptor
Main Volume Descriptor Sequence
9
Logical Volume Integrity Descriptor
Terminating Descriptor
11
Anchor Volume Descriptor Pointer
Logical Volume Integrity Sequence
First Anchor Point
メディアカード
400
16KB
Volume Structure
64MB
File Structure
420
16KB
Volume Structure
(上記と同じ構成)
440
12
File Set Descriptor
421
13
Terminating Descriptor
422
14
Space Bitmap
423
18
19
Reserved
424
31
File Structureの一部
File Entry, File Identifier Descriptor, Data

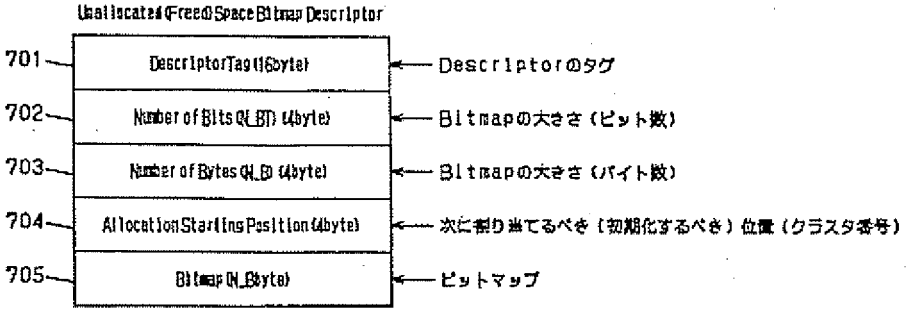
【図5】


501
Partition Descriptor
Partition Contents Use
500
Volume Structure
502
Partition Contents Use
520
File Structure
540
Volume Structure
Unallocated Space Table (8byte)
510
Unallocated Space Bitmap (8byte)
511
Partition Integrity Table (8byte)
512
Freed Space Table (8byte)
513
Freed Space Bitmap (8byte)
514
Reserved (88byte)
515
Unallocated Space Bitmap
ヘッダ(24byte)
503
ビットマップ
(セクタ数に応じた固定長)
504

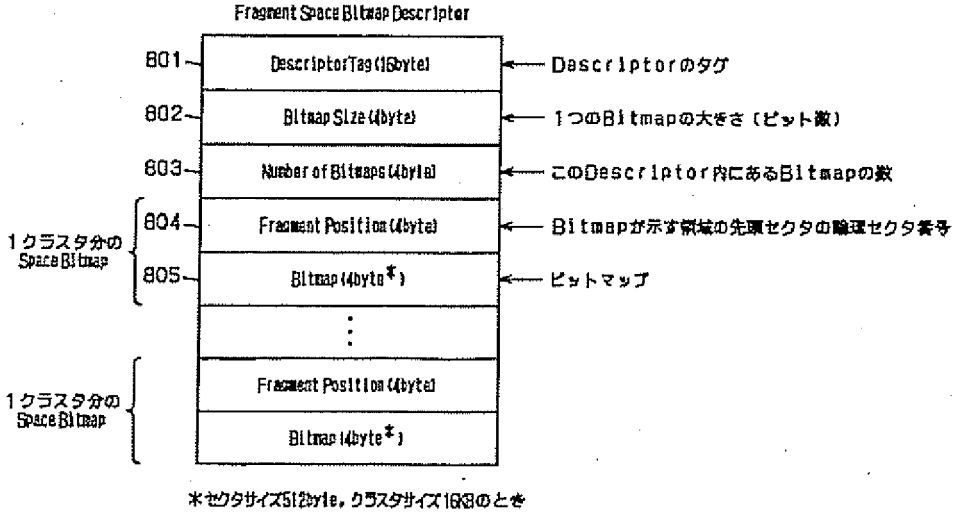
【図6】


Volume Structure
Partition Descriptor
File Structure
Volume Structure
Partition Contents Use
Unallocated Space Table (8byte)
610
Unallocated Space Bitmap (8byte)
611
Partition Integrity Table (8byte)
612
Freed Space Table (8byte)
613
Freed Space Bitmap (8byte)
614
Fragment Space Bitmap (8byte)
615
Reserved (88byte)
616
Unallocated Space Bitmap
ヘッダ(28byte)
ビットマップ：16KB単位
(セクタ数に応じた固定長)
631
Freed Space Bitmap
ヘッダ(28byte)
ビットマップ：16KB単位
(セクタ数に応じた固定長)
634
Fragment Space Bitmap
ヘッダ(24byte)
ビットマップ：512B単位
(可変長)
635

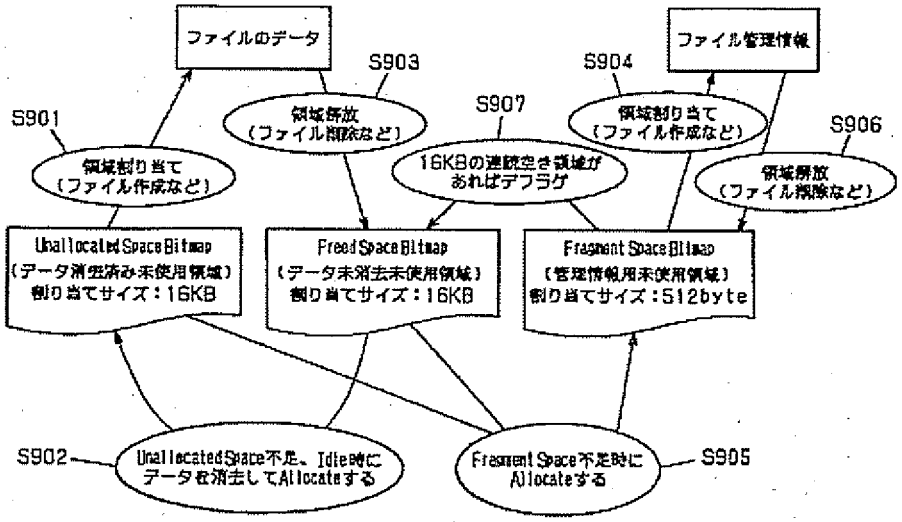
【図7】


Unallocated (Freed) Space Bitmap Descriptor
701
DescriptorTag (16byte)
Descriptorのタグ
702
Number of Bits (NUMBITS) (4byte)
Bitmapの大きさ（ビット数）
703
Number of Bytes (NUMBYTES) (4byte)
Bitmapの大きさ（バイト数）
704
Allocation Starting Position (4byte)
次に割り当てるべき（初期化するべき）位置（クラスタ番号）
705
Bitmap (N_Byte)
ビットマップ

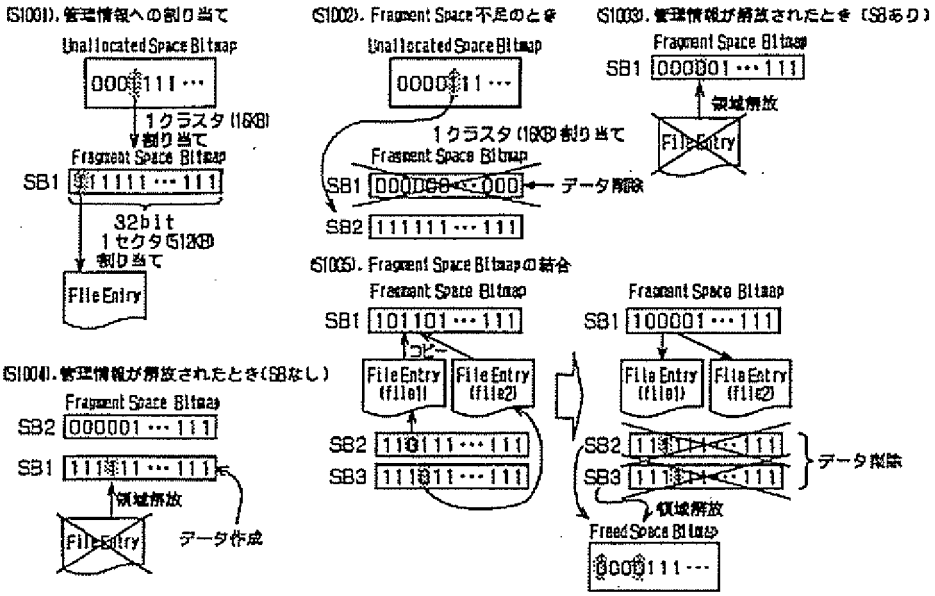
【図8】


Fragment Space Bitmap Descriptor
801
DescriptorTag (16byte)
Descriptorのタグ
802
Bitmap Size (4byte)
1つのBitmapの大きさ（ビット数）
803
Number of Bitmaps (4byte)
このDescriptor内にあるBitmapの数
1クラスタ分の Space Bitmap
804
Fragment Position (4byte)
Bitmapが示す領域の先頭セクタの論理セクタ番号
805
Bitmap (4byte*)
ビットマップ
Fragment Position (4byte)
Bitmap (4byte*)
*セクタサイズ512byte，クラスタサイズ16KBのとき

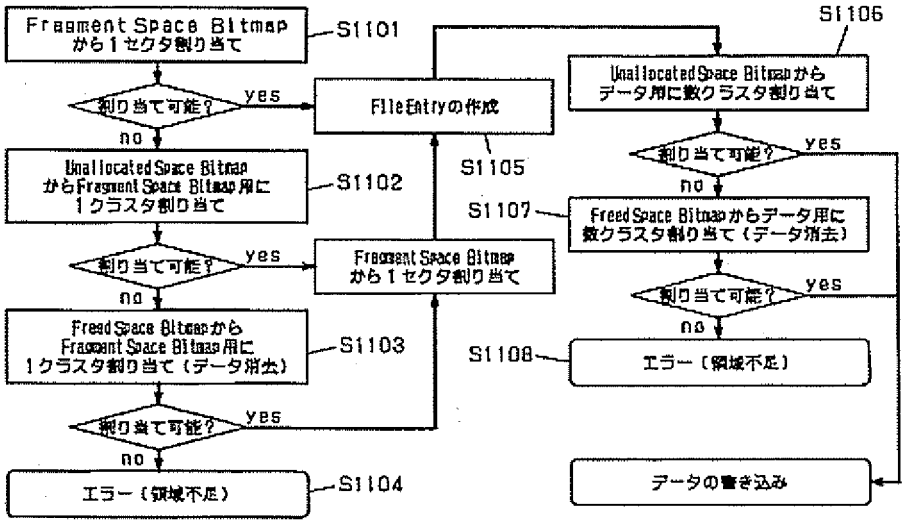
【図9】


ファイルのデータ
ファイル管理情報
S901
S903
S904
S906
S907
領域割り当て（ファイル作成など）
領域解放（ファイル削除など）
16KBの連続空き領域があればデフラグ
領域割り当て（ファイル作成など）
領域解放（ファイル削除など）
Unallocated Space Bitmap（データ消去済み未使用領域）割り当てサイズ：16KB
Freed Space Bitmap（データ未消去未使用領域）割り当てサイズ：16KB
Fragment Space Bitmap（管理情報用未使用領域）割り当てサイズ：512byte
S902
Unallocated Space不足、Idle時にデータを消去してAllocateする
Fragment Space不足時にAllocateする
S905

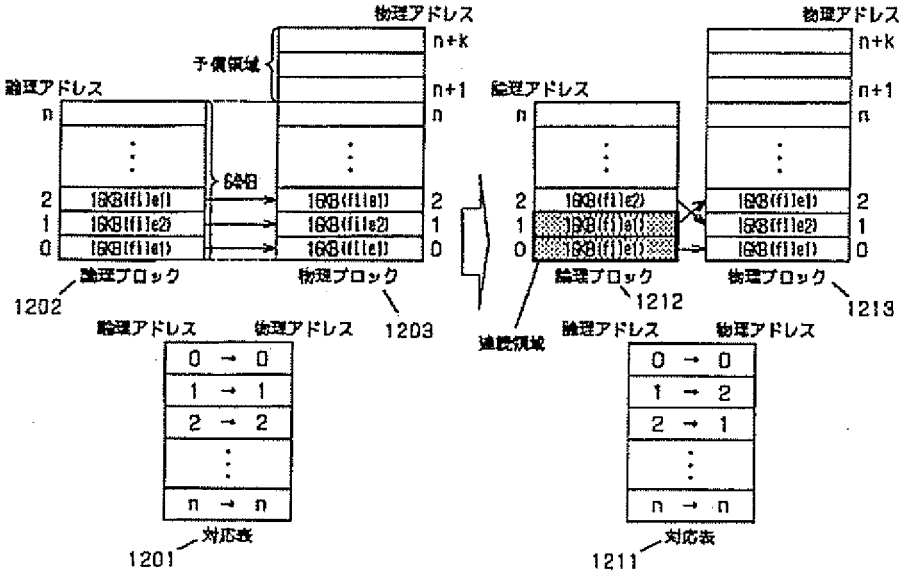
【図１０】


(S1001). 管理情報への割り当て
Unallocated Space Bitmap
000111…
1クラスタ(16KB)割り当て
Fragment Space Bitmap
SB1
11111…111
32bit
1セクタ(512B)割り当て
File Entry
(S1002). Fragment Space不足のとき
Unallocated Space Bitmap
0000111…
1クラスタ(16KB)割り当て
Fragment Space Bitmap
SB1
SB2
111111…111
データ削除
(S1003). 管理情報が解放されたとき(SBあり)
Fragment Space Bitmap
SB1
000001…111
領域解放
File Entry
(S1004). 管理情報が解放されたとき(SBなし)
Fragment Space Bitmap
SB2
000001…111
SB1
111111…111
領域解放
File Entry
データ作成
(S1005). Fragment Space Bitmapの結合
Fragment Space Bitmap
SB1
101101…111
コピー
File Entry (file1)
File Entry (file2)
SB2
110111…111
SB3
111011…111
Fragment Space Bitmap
SB1
100001…111
File Entry (file1)
File Entry (file2)
SB2
SB3
データ削除
領域解放
Freed Space Bitmap
000111…

【図１１】


Fragment Space Bitmap から1セクタ割り当て
S1101
割り当て可能?
yes
no
Unallocated Space Bitmap からFragment Space Bitmap用に 1クラスタ割り当て
S1102
割り当て可能?
yes
no
Freed Space Bitmapから Fragment Space Bitmap用に 1クラスタ割り当て(データ消去)
S1103
割り当て可能?
yes
no
エラー(領域不足)
S1104
File Entryの作成
S1105
Fragment Space Bitmap から1セクタ割り当て
S1106
Unallocated Space Bitmapから データ用に数クラスタ割り当て
割り当て可能?
yes
no
S1107
Freed Space Bitmapからデータ用に 数クラスタ割り当て(データ消去)
割り当て可能?
yes
no
S1108
エラー(領域不足)
データの書き込み

【図１２】



フロントページの続き

(72)発明者　榎　信行
大阪府門真市大字門真1006番地　松下電器産業株式会社内

(72)発明者　添田　純一郎
大阪府門真市大字門真1006番地　松下電器産業株式会社内

Ｆターム(参考)　5B035 AA00 BB09 BC05 CA29
5B058 CA26 KA33 YA20
5B060 AA02 AA06 AB26 AC11
5B082 AA11 AA13 DE04 EA01